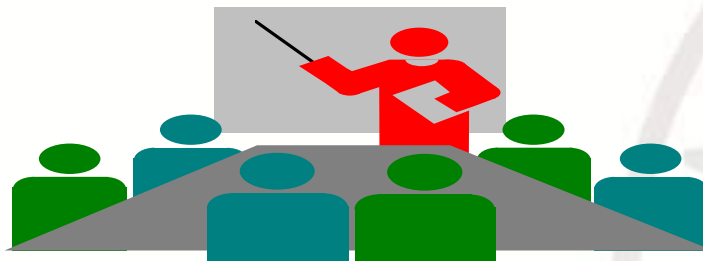




浙江大学  
ZHEJIANG UNIVERSITY



逻辑与计算机设计基础

# 逻辑与计算机设计基础实验 与课程设计

## 实验十一

### 寄存器堆 及寄存器传输设计

施青松

Asso. Prof. Shi Qingsong

College of Computer Science and Technology, Zhejiang University

[zjsqs@zju.edu.cn](mailto:zjsqs@zju.edu.cn)

# Course Outline



# 实验目的



1. 掌握寄存器结构及设计实现方法;
2. 掌握寄存器堆的工作原理及设计实现方法;
3. 掌握寄存器传输控制及设计实现方法;
4. 掌握基于总线的寄存器传输设计。
5. 了解计算机中寄存器及寄存器堆的概念



# 实验环境

## 实验设备

1. 计算机（Intel Core i3以上，1GB内存以上）系统
2. SWORD Kintex™-7 Board开发板
3. Xilinx ISE12.4及以上开发工具

## 材料

无



# Course Outline





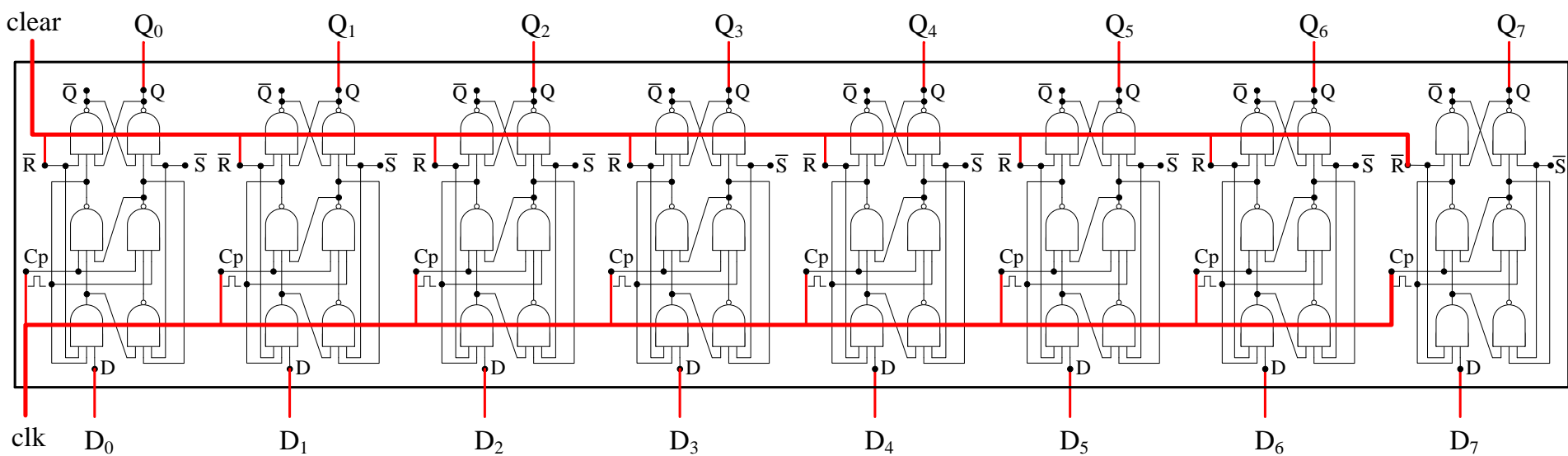
# 实验任务

1. 设计32位时钟写入的寄存器;
2. 设计 $8 \times 32\text{bits}$ 的寄存器堆(组);
3. 集成实验环境接口, 实现寄存器传输控制ALU运算。



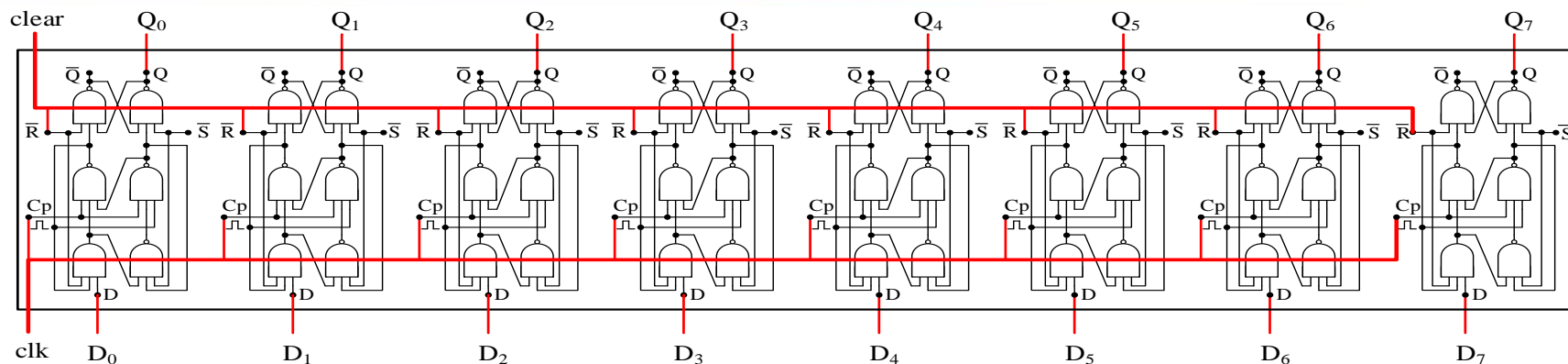
# Course Outline







# 8位边沿D触发器构成的基本寄存器结构化描述



```

module Regs_8bit (clk, D, clear, Q);
    // ...
wire[7:0] Qbar;
    assign cr = ~clear;
MB_DFF T0(.Cp(clk), .D(D[0]), .Rn(cr), .Sn(1'b1), .Q(Q[0]), .Qn(Qbar[0]) ),
    T1(.Cp(clk), .D(D[1]), .Rn(cr), .Sn(1'b1), .Q(Q[1]), .Qn(Qbar[1]) ),
    T2(.Cp(clk), .D(D[2]), .Rn(cr), .Sn(1'b1), .Q(Q[2]), .Qn(Qbar[2]) ),
    T3(.Cp(clk), .D(D[3]), .Rn(cr), .Sn(1'b1), .Q(Q[3]), .Qn(Qbar[3]) );
MB_DFF T4(.Cp(clk), .D(D[4]), .Rn(cr), .Sn(1'b1), .Q(Q[4]), .Qn(Qbar[4]) ),
    T5(.Cp(clk), .D(D[5]), .Rn(cr), .Sn(1'b1), .Q(Q[5]), .Qn(Qbar[5]) ),
    T6(.Cp(clk), .D(D[6]), .Rn(cr), .Sn(1'b1), .Q(Q[6]), .Qn(Qbar[6]) ),
    T7(.Cp(clk), .D(D[7]), .Rn(cr), .Sn(1'b1), .Q(Q[7]), .Qn(Qbar[7]) );
endmodule
    
```

# 锁存器

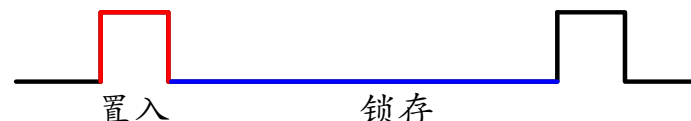
## ◎ 锁存器由多个一位锁存器/触发器并联构成

☞ 一般采用电平控制锁存:

☞ 高电平时置入数据

☞ 低电平时锁存数据, 为后续逻辑电路提供持续的信号

☞ 带使能端的RS触发器实现的锁存器



```
module lock(lock, D, clear, Q);          //Reset: clear=1
//...
assign cr    = {8{clear}} | (~D);
assign Din   = D & (~{8{clear}});
assign lock1 = lock | clear;
RS_EN  L0(.C(lock1), .R(cr[0]),.S(Din[0]), .Q(Q[0]), .Qn(Qbar[0])),
        L1(.C(lock1), .R(cr[1]),.S(Din[1]), .Q(Q[1]), .Qn(Qbar[1])),
        L2(.C(lock1), .R(cr[2]),.S(Din[2]), .Q(Q[2]), .Qn(Qbar[2])),
        L3(.C(lock1), .R(cr[3]),.S(Din[3]), .Q(Q[3]), .Qn(Qbar[3])),
        L4(.C(lock1), .R(cr[4]),.S(Din[4]), .Q(Q[4]), .Qn(Qbar[4])),
        L5(.C(lock1), .R(cr[5]),.S(Din[5]), .Q(Q[5]), .Qn(Qbar[5])),
        L6(.C(lock1), .R(cr[6]),.S(Din[6]), .Q(Q[6]), .Qn(Qbar[6])),
        L7(.C(lock1), .R(cr[7]),.S(Din[7]), .Q(Q[7]), .Qn(Qbar[7]));
endmodule
```

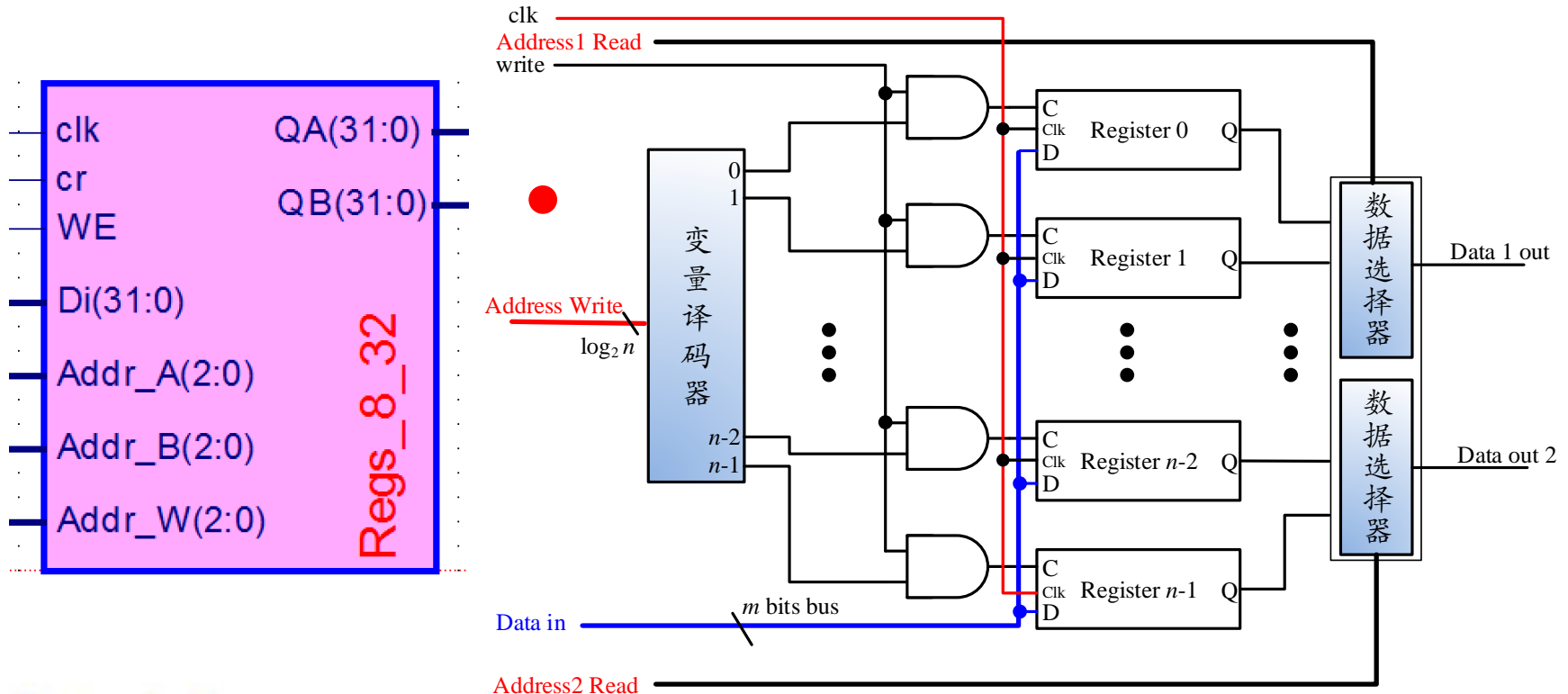
# 寄存器组 (Register files)

## ◎ 计算机基本部件 (Datapath 的一部分)

⌚ 多个寄存器的集合

◎ 寄存器写：寄存器地址 → 变量译码器

◎ 寄存器读：寄存器地址 → 数据选择器





# 8×32bits寄存器组硬件描述

```
module Regs_8_32(clk,cr,WE, Addr_W, Addr_A, Addr_B,Di, QA, QB);
// ...
    assign Y = ~Yi;
    assign CLK_R = {8{clk}};
    Reg_32 R0(CLK_R[0], Di, cr, Y[0], Do0 );
    Reg_32 R1(CLK_R[1], Di, cr, Y[1], Do1 );
    Reg_32 R2(CLK_R[2], Di, cr, Y[2], Do2 );
    Reg_32 R3(CLK_R[3], Di, cr, Y[3], Do3 );
    Reg_32 R4(CLK_R[4], Di, cr, Y[4], Do4 );
    Reg_32 R5(CLK_R[5], Di, cr, Y[5], Do5 );
    Reg_32 R6(CLK_R[6], Di, cr, Y[6], Do6 );
    Reg_32 R7(CLK_R[7], Di, cr, Y[7], Do7 );
    HTC138 D(.C(Addr_W[2]),.B(Addr_W[1]),.A(Addr_W[0]),.G(WE),.G_2A(1'b0),.G_2B(1'b0),
    .Y7(Yi[7]),.Y6(Yi[6]),.Y5(Yi[5]),.Y4(Yi[4]),.Y3(Yi[3]),.Y2(Yi[2]),.Y1(Yi[1]),.Y0(Yi[0]));
    MUX8T1_32 MUX_REGA(.I0(Do0), .I1(Do1), .I2(Do2), .I3(Do3), .I4(Do4), .I5(Do5),
    .I6(Do6), .I7(Do7), .s(Addr_A), .o(QA));
    MUX8T1_32 MUX_REGB( 寄存器堆输出B, 参考A写出 );
endmodule
```

时钟缓冲（并非最佳）：

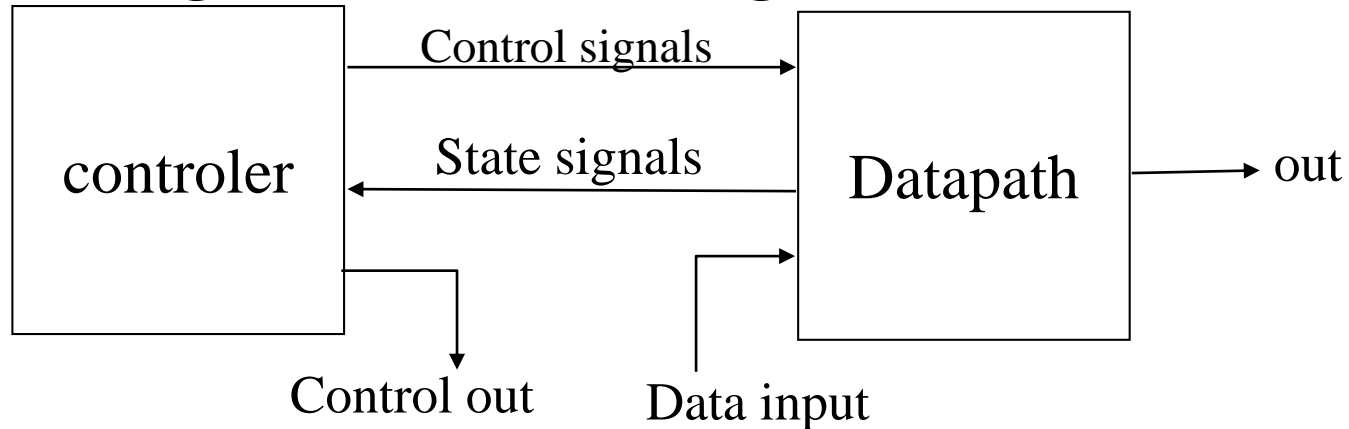
利用变量译码器使能端控制写入

工程时调用行为描述模块，学习时建议调用实验九原语描述D触发器

```
module Reg_32(cl, Di, cr, Load, Dot);
// ...
    reg [31:0] Q;
    always @(posedge clk or posedge clear)
        if(!clear) Q <= 0; else
            if (Load) Q <= D; else Q <= Q;
endmodule
```

# 寄存器传输操作

- **Register Transfer Operations** – The movement and processing of data stored in registers



- **Three basic components:**
  - Set of registers
  - Operations
  - control of operations
- **Elementary Operations** -load, count, shift, add, bitwise "OR", etc.
  - Elementary operations called *microoperations*

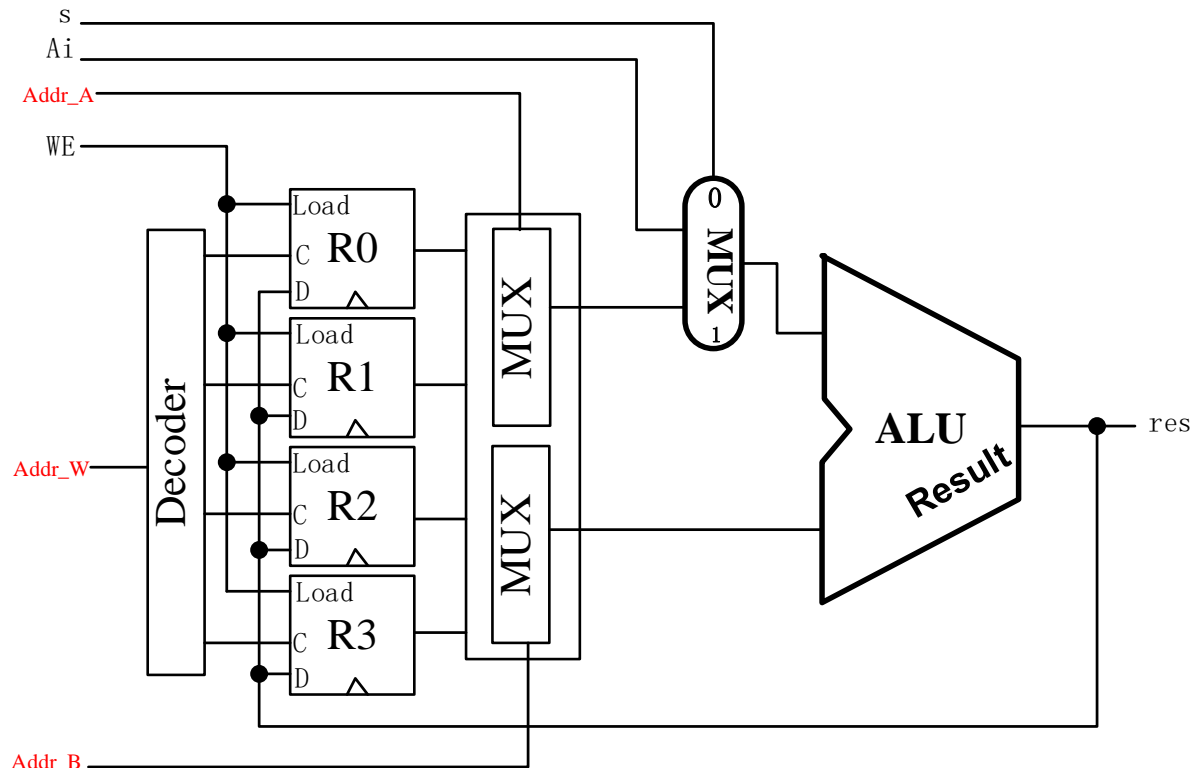
# 寄存器传输控制

◎ 运算微操作:  $R3 \leftarrow R1 + R2$

$W_1 W_0 \bar{A}_1 A_0 B_1 \bar{B}_0 WE s$ : ( $R2 \leftarrow R1 + R2$ )

◎ 加载微操作:  $R1 \leftarrow Ai + R0$

◎  $\bar{W}_1 W_0 \bar{B}_1 \bar{B}_0 WE \bar{s}$ : ( $R2 \leftarrow Ai + R0$ )



# Course Outline





# 设计工程一：Exp110-REGS

## ◎ 设计实现32位寄存器

- ⌚ 调用实验九MB\_DFF维持阻塞D触发器实现
- ⌚ 采用层次结构调用，行为和门级混合描述

## ◎ 设计实现8×32位寄存器堆(组)

- ⌚ 用结构化调用32位寄存器模块实现
  - ⊙ 1路写入：加载信号=WE、写地址=Addr\_W
  - ⊙ 2路读出：读地址=Addr\_A, Addr\_B

## ◎ 集成寄存器堆到实验十的“混合计算器”

- ⌚ 修改实验十顶层模块为：Calculation\_Exp110
- ⌚ 其余功能不变，新增功能：
  - ⊙ 寄存器堆与ALU和输入模块组成寄存器传输操作结构
  - ⊙ 初值用输入模块Ai，传输控制用输入模块Bi
  - ⊙ 寄存器堆输出显示用通道4、通道5



# 设计要点

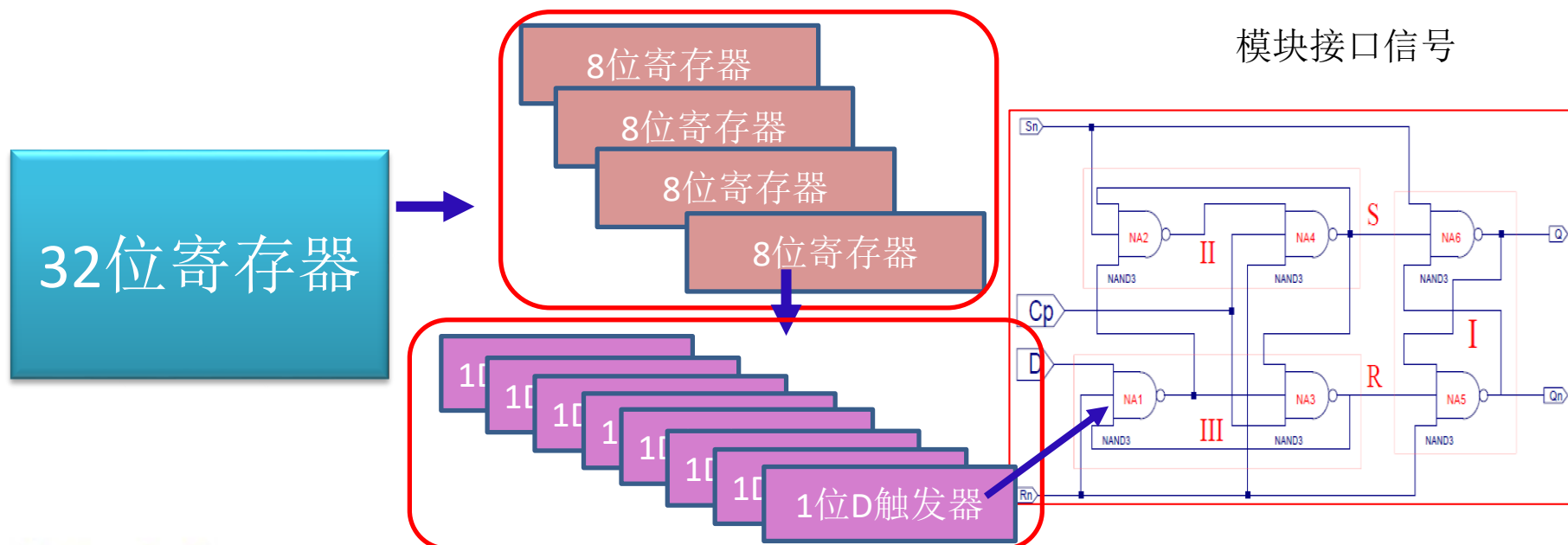
## ◎ 新建工程：Exp110-REGS

## ◎ 设计实现32位寄存器

- ☞ 建议模块名：Reg32.v
- ☞ 用Verilog HDL描述设计实现
- ☞ 层次调用实验九MB\_DFF 维持阻塞

```
module Regs_8_32(input clk,
                 input cr,
                 input WE,
                 input [2:0] Addr_W,
                 input [2:0] Addr_A,
                 input [2:0] Addr_B,
                 input [31:0] Di,
                 output [31:0] QA,
                 output [31:0] QB
                 );
```

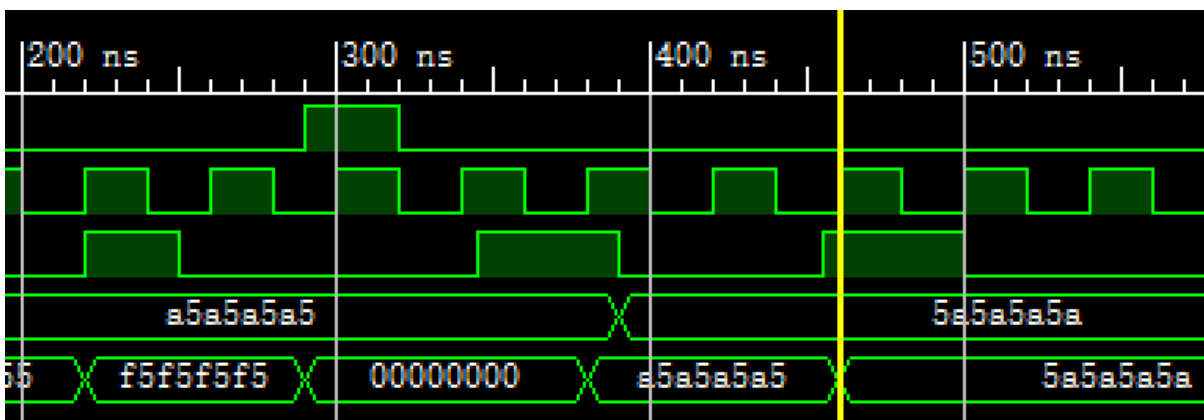
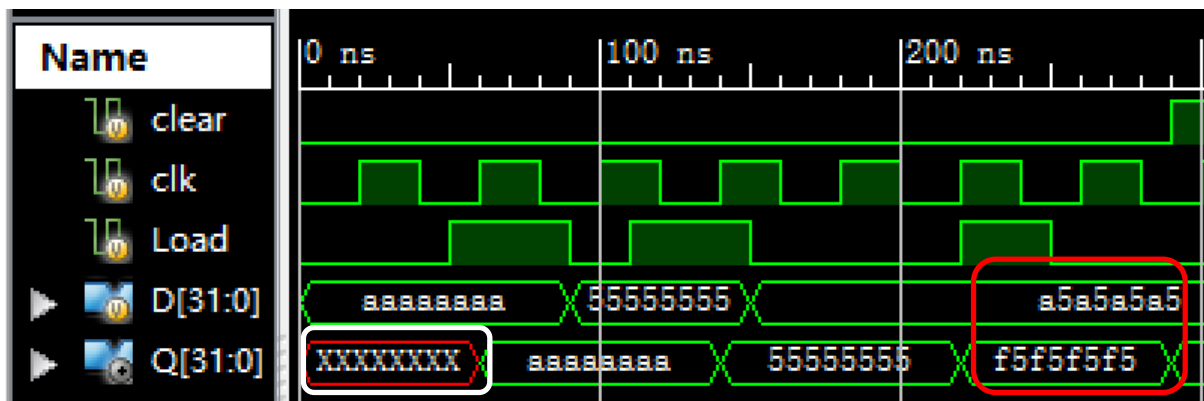
.....  
endmodule



# 32位寄存器激励与仿真

```

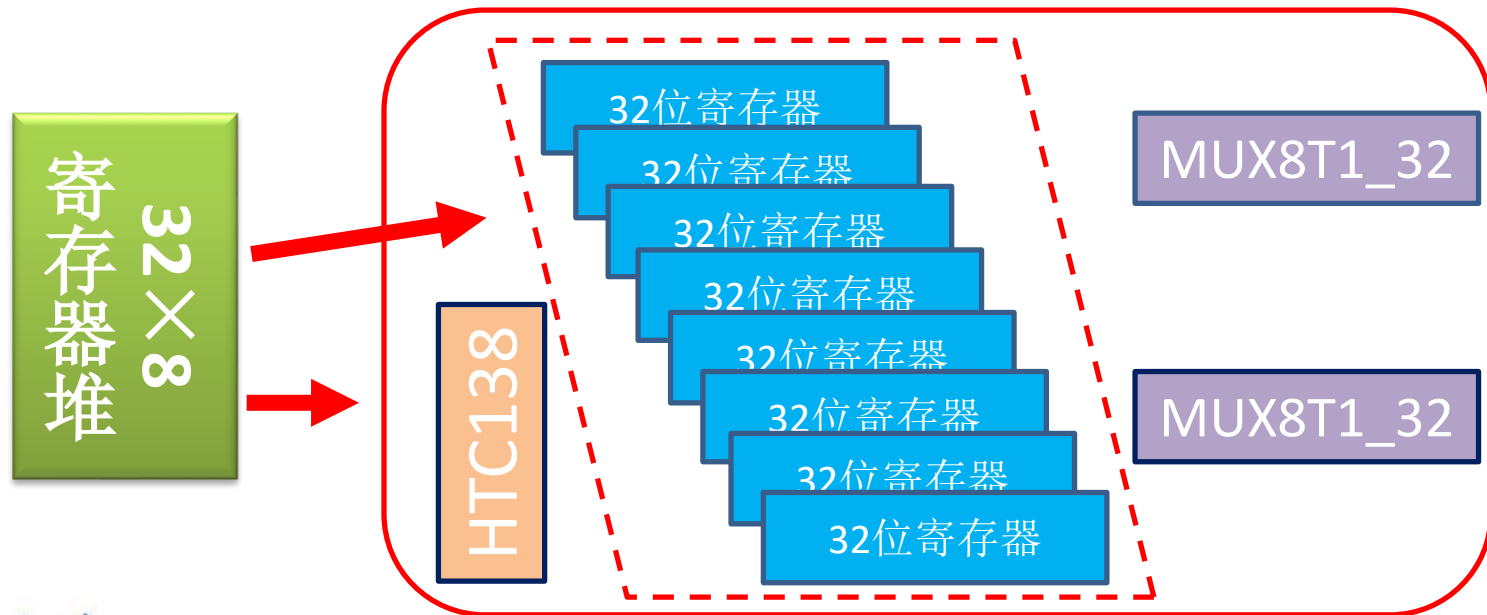
fork
  forever #20 clk <= ~clk;
  #20; clear = 0;
begin
  D = 32'hAAAAAAAA;
  #50; Load <=1;
  #40; Load <=0;
  D = 32'h55555555;
  #20; Load <=1;
  #40; Load <=0;
  D = 32'hA5A5A5A5;
  #70; Load <=1;
  #30; Load <= 0;
  #40; clear = 1;
  #30; clear = 0;
  #25; Load <=1;
  #45; Load <=0;
  D = 32'h5A5A5A5A;
  #65; Load <=1;
  #45; Load <=0;
end
join
end
  
```



## ◎ 设计实现 $8 \times 32$ 位寄存器堆(组)

☞ 用结构化调用32位寄存器模块实现

- 1路写入：写信号(Load)=WE、写地址=Addr\_W
- 2路读出：读地址=Addr\_A, Addr\_B





# 8×32寄存器堆激励参考

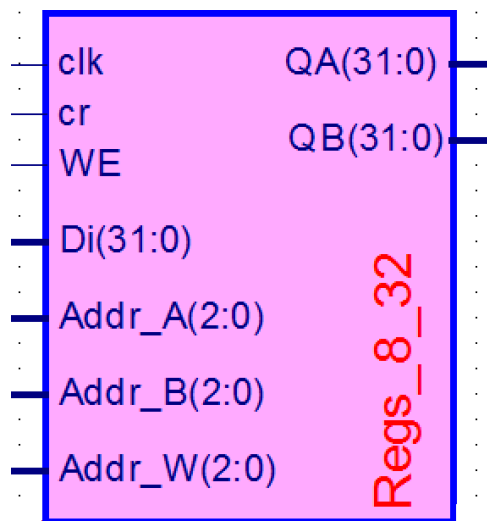
## ◎ 8个寄存器做遍历读写

Ⓔ 选择特征数据

⊙ AAAAAAAAA0+i

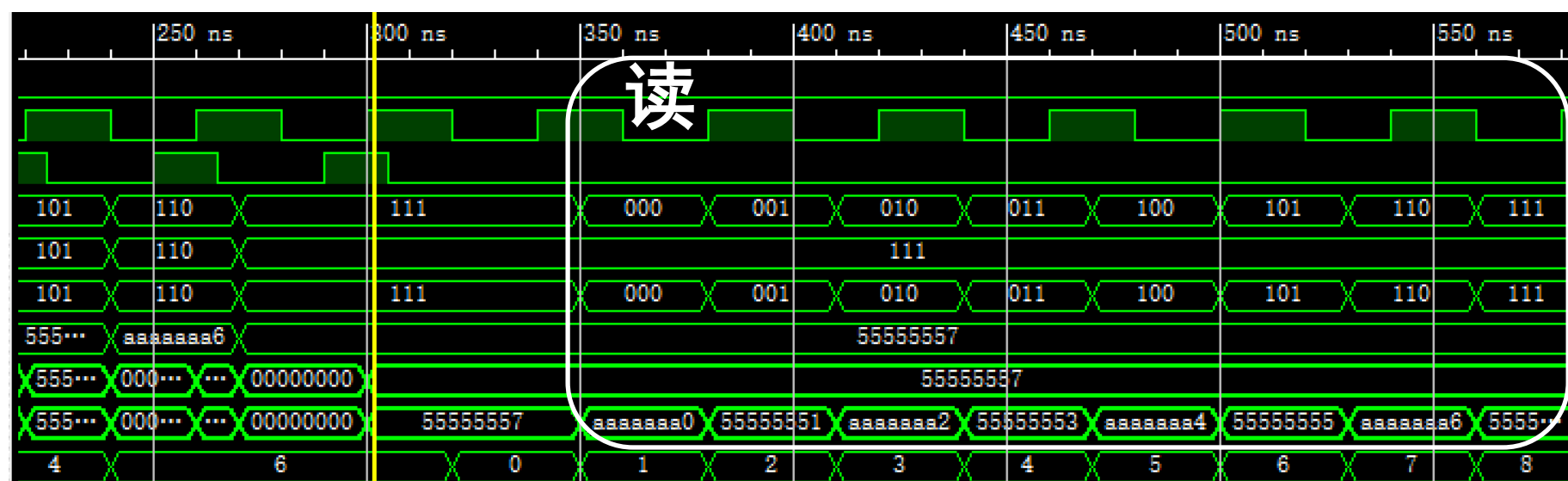
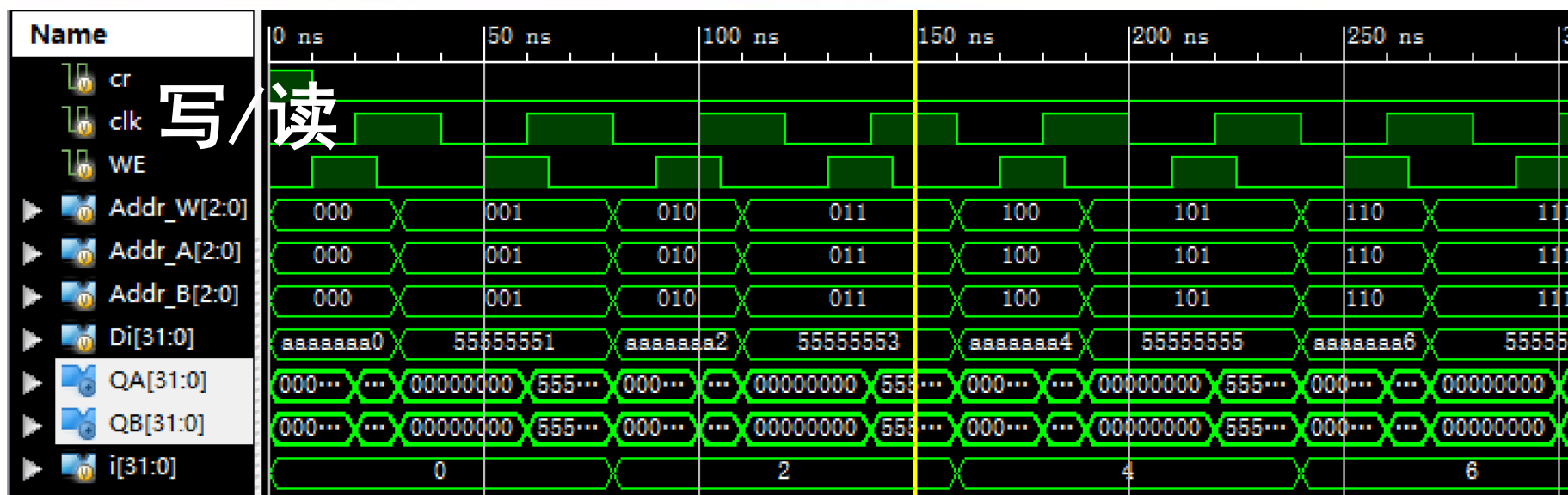
⊙ 55555550+i

## ◎ 仿真通过后制作逻辑符号



```
fork
  forever #20 clk <= ~clk;
  #10 cr = 0;
  begin
    for (i=0; i<8; i=i+2)begin
      Addr_W <= i;
      Addr_A <= i;
      Addr_B <= i;
      Di <= 32'hAAAAAAAA0+i;
      #10; WE <=1;
      #15; WE <=0;
      #5;
      Addr_W <= i+1;
      Addr_A <= i+1;
      Addr_B <= i+1;
      Di <= 32'h55555551+i;
      #20; WE <=1;
      #15; WE <=0;
      #15;
    end
    WE = 0;
    for (i=0; i<8; i=i+1)begin
      #30 Addr_W <= i;
      Addr_B <= i;
      Addr_B <= i;
    end
  end
```

# 8 × 32寄存器堆仿真图



# 集成混合计算器：增加ALU计算存储

## ◎ 集成Calculation

☞ 复制实验十的顶层模块，并改名为：Calculation\_Exp110.sch

⊙ 集成寄存器堆命名M11。输入模块Bi用于控制编程

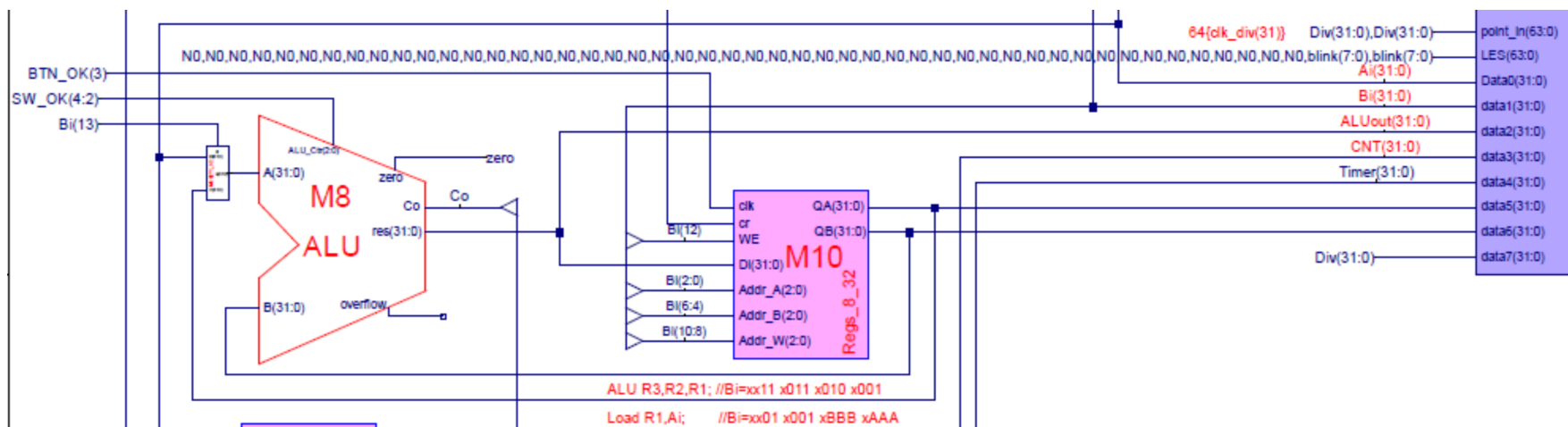
☞ 接口分配

⊙ 输入：clk=button\_out(3); WE=Bi(12); s=Bi(13); Addr\_A=Bi(2:0);

Addr\_B=Bi(6:4); Addr\_W=Bi(10:8);

⊙ ALU-B=QB; ALU-A=if(s) QA else Ai

⊙ 输出：QA→显示通道4(data4)，QB→显示通道5(data5)





# 辅助模块：MUX2T1\_32

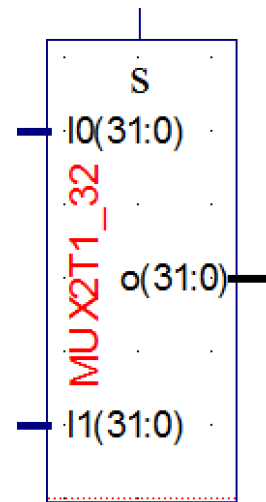
## ◎寄存器初值加载通道

- ☞ ALU-A输入通道 (DataPath) 选择
- ☞ 行为描述实现

```
module MUX2T1_32(input s,           //ALU A输入控制
                 input [31:0]I0,
                 input [31:0]I1,
                 output[31:0]o       //ALU A输入
                 );

    assign o = s ? I1 : I0;

endmodule
```



## ◎通道控制

- ☞ s=输入模块Bi(13)
- ☞ s=1, ALU A输入Ai
- ☞ s=0, ALU A输入Q



# 寄存器传输控制：ALU运算编程

## ◎ 数据通路(DataPath) 控制器编程

☞ 模拟问题设计控制模块

⊙ Verilog HDL实现

手工编程输入模块Bi控制格式

⊙ ROM实现

⊙ 手工编程实现

31	15	控制12	11	写地址8	7	读地址4	3	读地址0
			xx s WE	xAddr_W	xAddr_B	xAddr_A		

## ◎ 手工编程

☞ 采用输入模块Bi输入数据控制ALU运算的通路

⊙ 寄存器堆清零：长按复位键，第一次运算前必须清零

⊙ 加载数据到Reg：输入模块Ai+0(R0)→Rx **R0清零!!!**

◆ Load R1, Ai; //R1←Ai, Bi=xx01 x001 x000 xAAA

⊙ ALU寄存器之间运算：

**x任意值**

◆ ALU-OP R3,R2,R1; // R3←R1 OP R2,

Bi=xx11 x011 x010 x001



# 物理验证

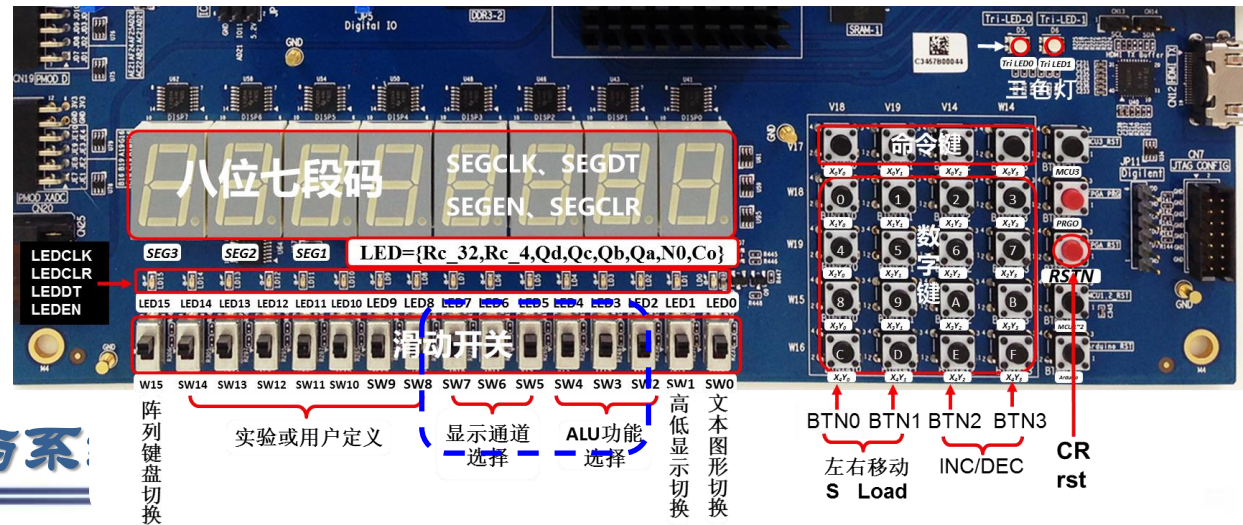
## □ UCF引脚定义

### □ 输入

- SW[7:5]=通道选择
  - =000: 修改被加数、=001: 修改加数、=010: ALU输出、=011:32位计数输出
- SW[4:2]=ALU功能控制
- SW[1]= 高低16位数据选择
- SW[15]=0独立按键， SW[15]=1阵列键盘
- BTN0、BTN1=左右移动； BTN1、BTN3=输入修改

### □ 输出: LED={Rc\_32,Rc\_4,Qd,Qc,Qb,Qa,N0,Co}

### □ 同实验十





# 输入设备功能定义

开关定义	=0	=1	备注
SW[0]	未用		
SW[1]	32位二进制高16位	32位二进制低16位	
SW[4:2]	ALU功能选择		参考ALU功能表
SW[7:5]	通道选择 =000 =001 =010 =011 <b>=100</b> <b>=101</b> =110 =111	通道0 通道1 通道2 通道3 <b>通道4</b> <b>通道5</b> 通道6 通道7	<b>Ai</b> <b>Bi</b> <b>RES(ALU_Out)</b> <b>Cnt</b> <b>寄存器A输出QA</b> <b>寄存器B输出QB</b>

按键定义	=0	=1	备注
Button[0]	左移	右移	<b>移动方向控制</b>
Button[1]		<b>正脉冲移动</b>	
Button[2]		<b>正脉冲输入修改</b>	<b>递增修改</b>
Button[3]		Regs单步时钟	长按复位

# ALU及寄存器功能控制

ALU功能选择	=0 (XXX)	=1 (功能)	备注
SW[4:2]= ALU_Ctr(2:0)	000	与	 <p>SW[4]=0</p> <p>SW[4]=1 A&lt;B, A=1, 否则A=0</p>
	001	或	
	010	加	
	011	自定义	
	100	自定义	
	101	自定义	
	110	减	
	111	Slt	
Button[3]=clk	寄存器单步时钟		长按复位
输入模块Bi[2:0]			寄存器堆A口读地址
输入模块Bi[6:4]			寄存器堆B口读地址
输入模块Bi[10:8]			寄存器堆写地址
输入模块Bi[12]	禁止写入	写入	寄存器堆写控制
输入模块Bi[13]	ALU A = Ai	ALU A = QA	ALU B输入源选择





# ALU存储计算

## ◎ 任选一非常简单计算

- ☞ 手工编程单步运行
- ☞ 用Verilog HDL设计控制模块运行(选修)
- ☞ 用ROM控制实现(选修)

## ◎ 参考例子：SW[4:2]=010,ALU做加法

- ☞ 输入常数“1”累加：  $A_i=1$

$R1 \leftarrow 1$             //Bi=xx01 x001 x000 xxxx

$R2 \leftarrow R1+R1$     //Bi=xx11 x010 x001 x001

$R3 \leftarrow R2+R2$     //Bi=xx11 x011 x010 x010

$R4 \leftarrow R3+R3$     //Bi=xx11 x100 x011 x011

$R5 \leftarrow R4+R4$     //Bi=xx11 x101 x100 x100

.....



# 思考题

---

◎ 不通过 $A_i$ 输入常数“1”如何实现1的累加？



同学们：每次做完实验请整理好实验台，放好  
仪器，理清桌面。



Thank you!

